Danial Zohourian LLM Text Detection Report

I implemented three distinct models for a Language Model (LLM) Text Detection task. The objective was to explore various approaches and assess their performance in identifying and understanding textual information within a given context.

To make the models more effective, I realized the initial dataset was quite small. So, I decided to add more data from different sources to give the models a better grasp of different text patterns. This helped prevent the models from becoming too specialized and improved their accuracy in identifying text in various situations. By using a larger and more diverse dataset, I aimed to make the models more adaptable and reliable for LLM Text Detection.

1. DeBERTaV3 Pretrained + TF-IDF Model

The first model utilized a DeBERTaV3 pretrained model in conjunction with a TF-IDF (Term Frequency-Inverse Document Frequency) approach. The synergy of these two techniques aimed to capitalize on the contextual understanding provided by DeBERTaV3, complemented by the statistical relevance captured through TF-IDF weighting. The model was trained over 10 epochs to refine its ability to detect and interpret text patterns.

Model Architecture

The model is initialized by loading the pre-trained DeBERTa model and tokenizer, modifying the classifier heads for both DeBERTa and TF-IDF, and combining them with a larger linear layer.

Components

1. **DeBERTa Component:**
   - The DeBERTa model is loaded using **DebertaModel.from_pretrained**.
   - The tokenizer is initialized using **AutoTokenizer.from_pretrained**.
   - The DeBERTa classifier head is a linear layer with an output size of 256.
2. **TF-IDF Component:**
   - The TF-IDF classifier head is a linear layer with an output size of 128.
3. **Combined Classifier:**
   - A final linear layer combines the outputs of both DeBERTa and TF-IDF classifiers. The input size is the sum of their output sizes (128 + 256).

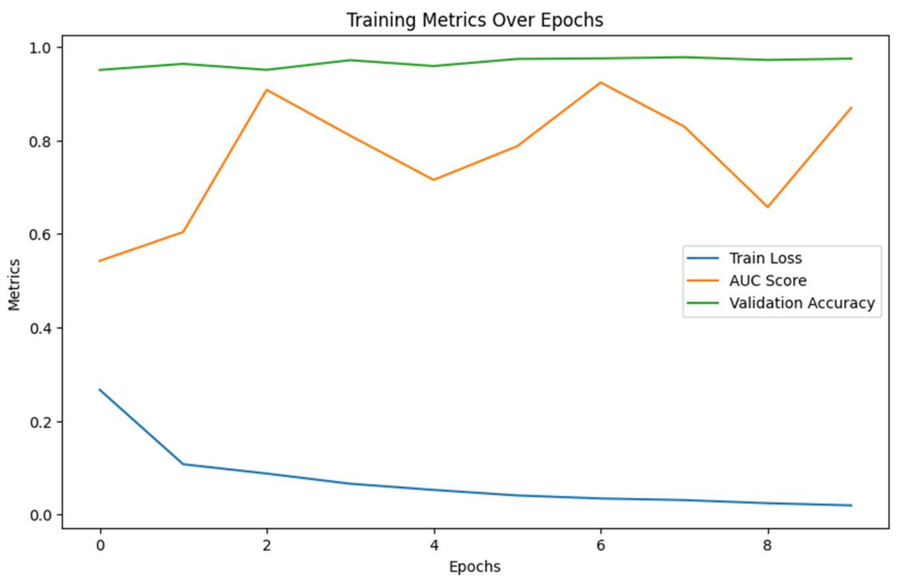
Forward Pass

The forward pass involves the following steps:

1. **DeBERTa Forward Pass:**
   - Input is fed through the DeBERTa model.
   - The output embeddings are obtained, and attention values are stored.
   - The DeBERTa classifier head produces logits.

2. **TF-IDF Forward Pass:**
    - Input is fed through the TF-IDF classifier head, producing TF-IDF logits.
3. **Combining Embeddings:**
    - DeBERTa and TF-IDF logits are concatenated along the feature dimension.
4. **Combined Forward Pass:**
    - The combined embeddings are passed through the final linear layer to produce the binary classification logits.

The model was trained over 10 epochs to refine its ability to detect and interpret text patterns.



Overall Trends:

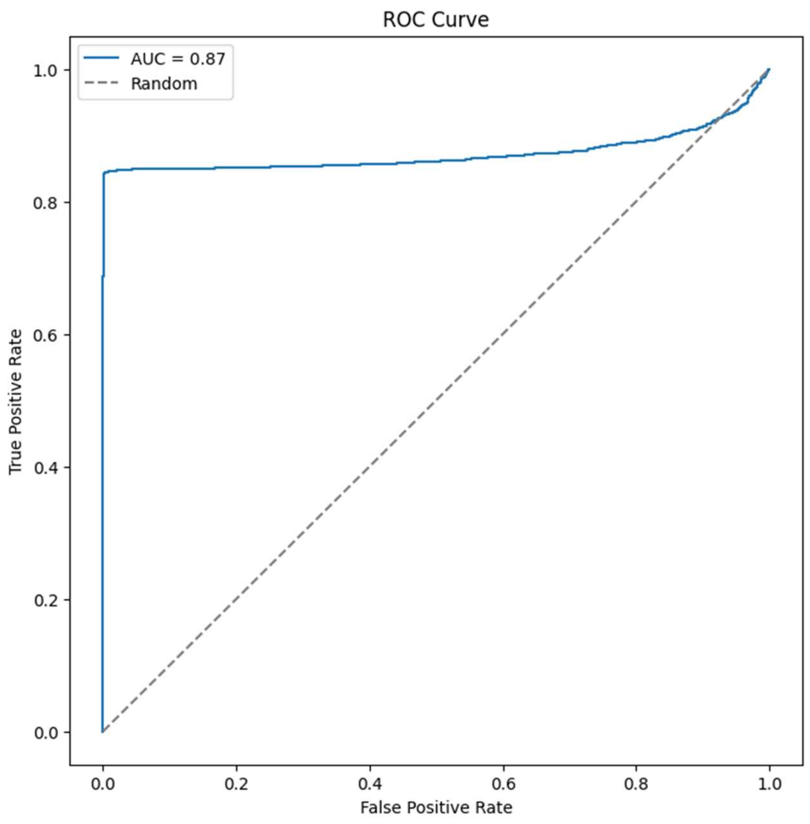
- The model generally exhibits good convergence with decreasing training loss.
- Validation accuracy is consistently high, indicating the model's effectiveness in predicting the correct class.
- AUC scores vary across epochs, suggesting fluctuations in the model's ability to distinguish between positive and negative instances.

Test Set Performance:

After training the model for multiple epochs, evaluating its performance on a separate test set provides a crucial assessment of its generalization capabilities.

- **Test Accuracy:** 97.49%
    - The test accuracy is high, indicating that the model performs well on unseen data. It correctly classifies approximately 97.49% of instances in the test set.
- **Test AUC Score:** 87.25%

- The test AUC score is also strong, suggesting that the model maintains a good balance between sensitivity and specificity on the test data. The AUC score of 87.25% indicates the model's ability to discriminate between positive and negative instances.



In my perspective, the AUC score could have potentially reached even higher values if the model had more training time over a greater number of epochs. Given the limited training time of 10 epochs, the model might not have fully converged to an optimal state. The ROC score, which measures the trade-off between true positive rate and false positive rate, could potentially grow higher with additional training epochs. This is because the model might uncover more nuanced patterns and better optimize its decision boundaries with extended exposure to the training data.

Comparison with Validation Set:

- The test accuracy is consistent with the high validation accuracies observed during training.
- The test AUC score is slightly lower than the best validation AUC score but remains at a commendable level.

Overall Assessment:

The test set results affirm the model's ability to generalize well to new, unseen data. The high accuracy and AUC score indicate that the model has learned meaningful patterns from the training data, and it successfully applies this knowledge to make accurate predictions on the test set. This robust performance on the test set reinforces the model's reliability for the text classification task.

Explainability:

The **attention_explain** function provides insights into which parts of the input have received higher attention during a specific layer of the model. It tokenizes the input, calculates normalized attention weights, and identifies sentences with higher cumulative attention.

The function is particularly useful for interpreting attention mechanisms in natural language processing models, helping to understand which words or phrases are deemed more important during a specific layer's processing.

After feeding the function the following human-written text collected from Wikipedia, the model output **[2.5548, -2.4132]**, suggests that the provided input is not likely generated by a Language Model. The values in the tensor represent scores for different classes, and the higher value (2.5548) is associated with the negative class (human-written), indicating a lower likelihood of being LLM-generated.

NYT Text:

"The New York Times is a national daily newspaper based in New York City. A newspaper of record, it is the second-largest newspaper by circulation and one of the longest-running newspapers in the United States."

The sentence "A newspaper of record, it is the second-largest newspaper by circulation and one of the longest-running newspapers in the United States" has the highest cumulative attention (1.565952). This suggests that the model paid significant attention to this sentence when making the classification decision, and it likely influenced the negative outcome.

On the other hand, the sentence "The New York Times is a national daily newspaper based in New York City" has a lower cumulative attention (0.782702). Despite being part of the input, this sentence appears to have contributed less to the negative classification.

After providing the initial human-written text to ChatGPT and instructing it to rewrite the text, the higher value (2.0225) is now associated with the positive class, suggesting that the rewritten text is more likely to be LLM-generated.

ChatGPT Text:

"Based in New York City, The New York Times is a national daily newspaper renowned as a newspaper of record. It holds the position of the second-largest newspaper by circulation and stands as one of the longest-running newspapers in the United States."

The sentence "It holds the position of the second-largest newspaper by circulation and stands as one of the longest-running newspapers in the United States" has the highest cumulative attention (1.609411). This indicates that the model assigned significant attention to this sentence during the positive classification decision.

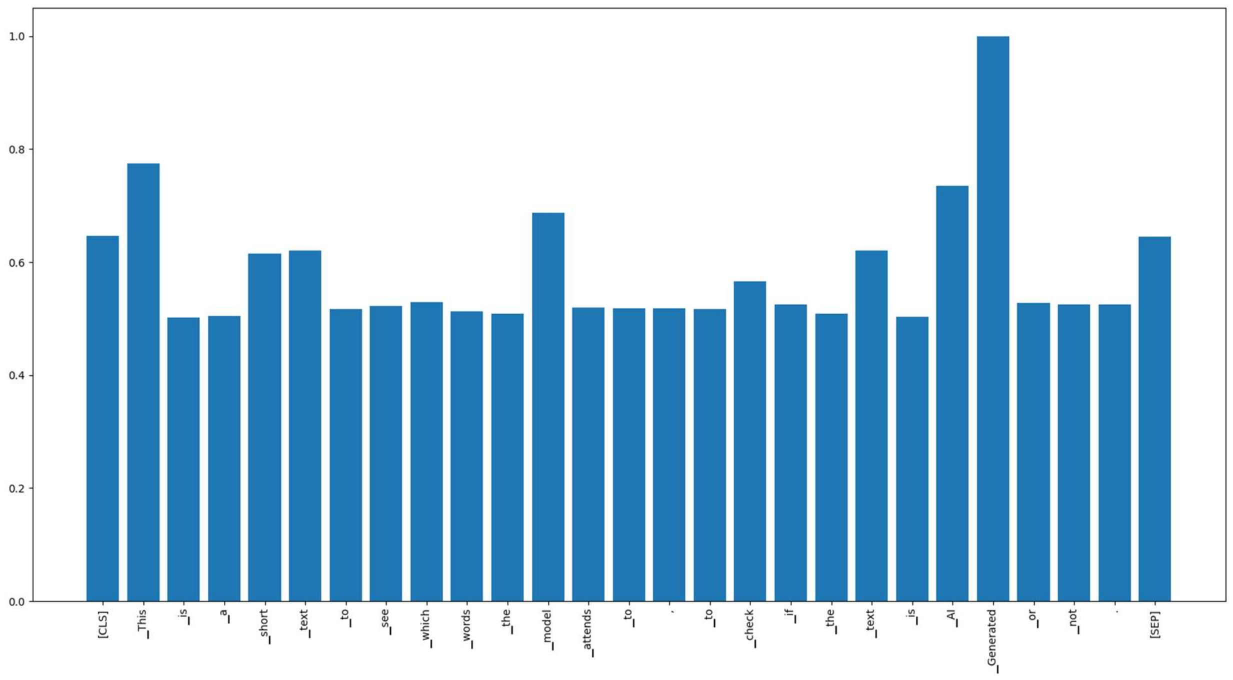
The introductory sentence "Based in New York City, The New York Times is a national daily newspaper renowned as a newspaper of record" also has substantial cumulative attention (0.804392), contributing to the positive classification.

The **visualize_attention** function calculates the attention weights for a specific layer (**layer_num**) in a neural network. It computes the maximum attention value across attention heads for each token, resulting in a vector of attention weights corresponding to each token in the input.

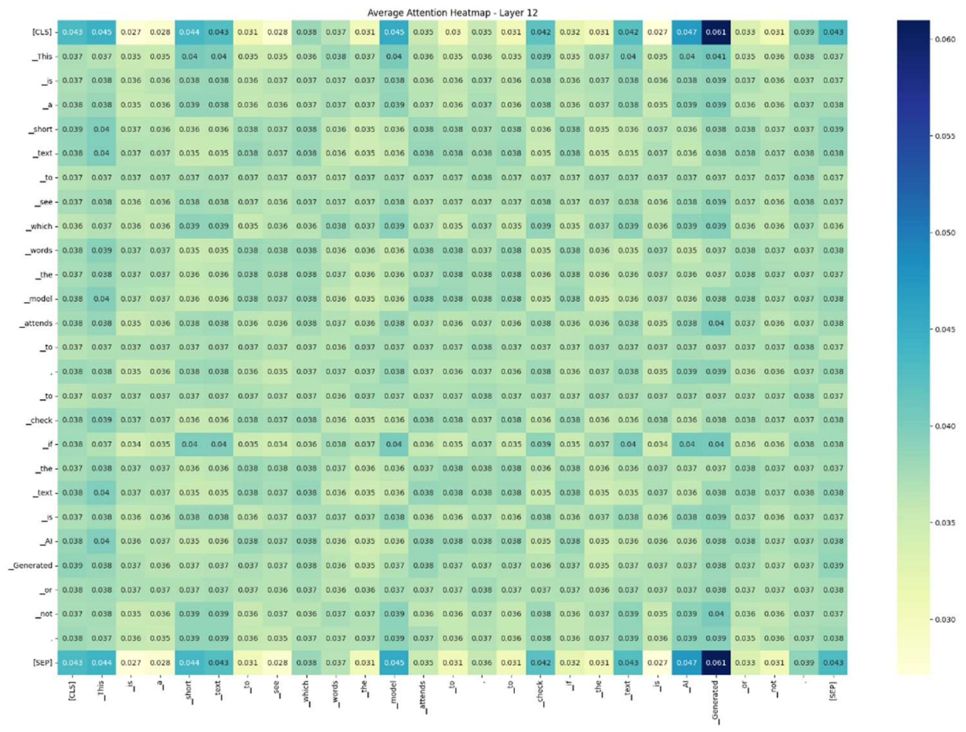
Given the following text, we can see which words the model paid more attention to, to guess the outcome of the classification, which is human-written.

Short Text:

"This is a short text to see which words the model attends to, to check if the text is AI Generated or not."



I also wrote a code that generates and visualizes an attention heatmap for a specific layer (**layer_num**) in a neural network. The heatmap provides insights into how different tokens in the input sequence interact with each other, capturing the attention distribution.



2. Multinomial Naive Bayes

The second model employed a Multinomial Naive Bayes algorithm. This classic probabilistic model is known for its simplicity and efficiency in text classification tasks. Its implementation in the LLM Text Detection task was explored to evaluate its performance in discerning textual features without the complexity introduced by deep neural networks.

Precision, Recall, and F1-Score:

- **Precision:**
    - Class 0 (Negative class): 93%
    - Class 1 (Positive class): 95%

Precision represents the accuracy of the positive predictions made by the model. In this context, it indicates the percentage of correctly identified positive instances out of all instances predicted as positive.

- **Recall:**
    - Class 0: 92%
    - Class 1: 95%

Recall, also known as sensitivity or true positive rate, measures the ability of the model to capture all positive instances. In this scenario, it denotes the percentage of correctly identified positive instances out of all actual positive instances.

- **F1-Score:**
    - Class 0: 92%
    - Class 1: 95%

The F1-score is the harmonic mean of precision and recall, providing a balanced measure of a model's overall performance. It is particularly useful when there is an imbalance between the classes.

Support:

- Class 0: 1769 instances
- Class 1: 2768 instances

Support indicates the actual number of instances in each class within the dataset.

Accuracy:

- Overall Accuracy: 94%

Accuracy is the ratio of correctly predicted instances to the total instances. In this case, the model achieved an accuracy of 94%, showcasing its effectiveness in making correct predictions across both classes.

Macro and Weighted Averages:

- **Macro Avg:**
    - Precision: 94%

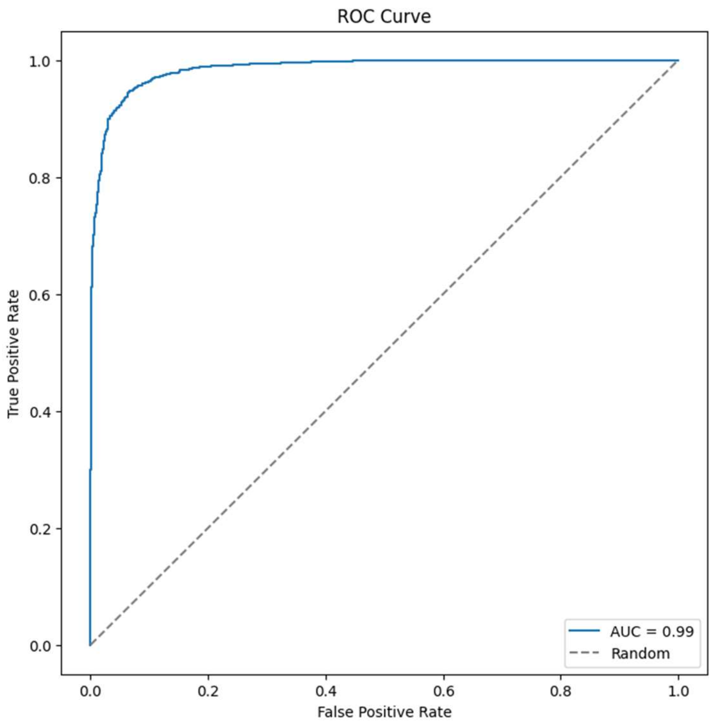
    - Recall: 94%
    - F1-Score: 94%

The macro average calculates the metrics independently for each class and then takes the average. It treats all classes equally.

- **Weighted Avg:**
    - Precision: 94%
    - Recall: 94%
    - F1-Score: 94%

The weighted average considers the number of instances in each class, providing a more balanced representation when there is a class imbalance.

The AUC-ROC score is also high with this model:



In summary, the Multinomial Naive Bayes model demonstrates high precision, recall, and F1-score, with an ROC-AUC score of 99%. These results indicate its effectiveness in accurately classifying instances in the LLM Text Detection task.

3. TF-IDF Neural Network

The third model involved the development of a TF-IDF Neural Network, combining the principles of TF-IDF weighting with a neural network architecture. This approach aimed to leverage the strengths of both techniques, using the neural network to capture intricate patterns and nuances present in textual data while maintaining the interpretability facilitated by TF-IDF weighting. The model underwent training for 10 epochs to enhance its proficiency in text detection.

Precision, Recall, and F1-Score:

- **Precision:**
    - Class 0 (Negative class): 98%
    - Class 1 (Positive class): 99%
- **Recall:**
    - Class 0: 99%
    - Class 1: 99%
- **F1-Score:**
    - Class 0: 98%
    - Class 1: 99%

Support:

- Class 0: 1769 instances
- Class 1: 2768 instances

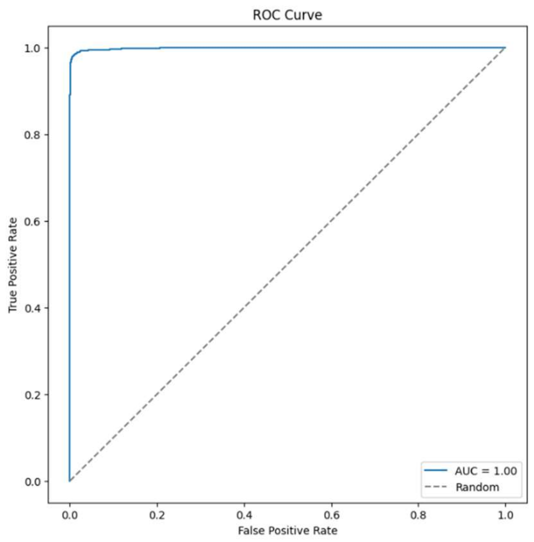
Accuracy:

- Overall Accuracy: 99%

Macro and Weighted Averages:

- **Macro Avg (Macro Average):**
    - Precision: 98%
    - Recall: 99%
    - F1-Score: 98%
- **Weighted Avg (Weighted Average):**
    - Precision: 99%
    - Recall: 99%
    - F1-Score: 99%

The AUC-ROC score is higher with this model:



In summary, the TF-IDF Neural Network demonstrates outstanding precision, recall, and F1-score, with an overall accuracy of 99%. These results indicate its remarkable effectiveness in accurately classifying instances in the LLM Text Detection task, highlighting its potential as a robust text detection model.

In the end, I added another human-written text and saw how the 2nd and 3rd networks work on them.

Tetris Text:

"Nearly all Tetris games allow the player to press a button to increase the speed of the current piece's descent or cause the piece to drop and lock into place immediately, known as a "soft drop" and a "hard drop", respectively. While performing a soft drop, the player can also stop the piece's increased speed by releasing the button before the piece settles into place. Some games allow only one of either soft drop or hard drop; others have separate buttons for each. Many games award a number of points based on the height that the piece fell before locking, so using the hard drop generally awards more points."

TF-IDF Results:

NYT Text: LLM - Incorrect

ChatGPT Text: LLM - Correct

Short Text: Human - Correct

Tetris Text: LLM - Incorrect

MultinomialNB Results:

NYT Text: Human - Correct

ChatGPT Text: Human - Incorrect

Short Text: Human - Correct

Tetris Text: Human - Correct